Specification Amendments

Please replace paragraph [0002] with the following amended paragraph:

[0002] This application is a Continuation of U.S. Patent Application Serial No. 09/923,587, filed August 7, 2001, ~~entitled "A Method for Calibrating a Color Marking Engine for Halftone Operation,"~~ now U.S. Patent No. 6,636,326, which is a Continuation of U.S. Patent Application Serial No. 09/229,242, now U.S. Patent No. 6,271,937, ~~issued August 7, 2001,~~ which is a ~~Continuation-in-Part~~ Division of U.S. Patent Application Serial No. 08/788,113 ~~09/229,242~~, filed ~~January 12, 1999~~ January 23, 1997, now U.S. Patent No. 6,035,103, ~~entitled "Color Correction of Dot Line Linearities in Multiple Print Engine System", (Atty. Docket No. TRSY-24,508)~~ which is a Continuation-in-Part of U.S. Patent Application Serial No. 08/698,999, filed August 16, 1996, now U.S. Patent No. 5,859,711, ~~entitled "Multiple Print Engine with Virtual Job Routing", (Atty. Docket No. TRSY-23,684)~~ which is a Continuation-in-Part of U.S. Patent Application Serial No. 08/511,641, filed August 7, 1995, now U.S. Patent No. 6,657,741, ~~entitled "Virtual Single Print Engine with Software Rip", (Atty. Docket No. TRSY-23,617)~~ and is related to U.S. Patent Application Serial No. 08/180,636, filed January 13, 1994, ~~and entitled "Multiple Printer Module Electrophotographic Printing Device" (Atty. Dkt. No. TRSY-22,391)~~ now U.S. Patent No. 5,596,416.

Please replace paragraph **[0031]** with the following amended paragraph:

[0031] Referring now to FIGURE 6, there is illustrated a flowchart depicting the operation for a duplex print job. In the flowchart of FIGURE 6, a face up output is considered which is initiated at a block 260. The function block then flows to a decision block 262 to determine if the value of N is even. If so, the program flows to a function block 264 to print the jobs N-2, N-4, . . ., 2 [[-2, -4 . . ., 2]]. The program then flows to a decision block 266, which determines whether the value of N is odd. However, if N was odd at decision block 266, the program would flow along the "N" path to the output of the decision block 266 and then to a function block 268 to print the N + 1 copies and blank copies and then print the N - 1, N - 3, ... 1 pages. The flowchart would then flow to a function block 270. It is noted that if N is even at decision block 266, the program

would flow to the function block 270. Function block 270 is a function block wherein a user ~~annually~~ manually turns the output stack 180° without flipping the stack and then puts it back in the drawer of the printer from which it came. The program then flows to a decision block 74 to determine if the value of N is even, and if so, to the function block 270 along the "Y" path to print the pages 1, 3, 5, . . ., N-1 ~~1, 3, 5, ... 1~~, and then to a decision block 278 to determine if the value of N is odd. The program at this point will flow along the "N" path to a N block 280. However, if the value of N is determined to be odd at decision block 274, the program will flow through the output of decision block 278 and to the input of a function block 282 which will print the pages 1, 3, 5, . . . N.

Please replace paragraph **[0032]** with the following amended paragraph:

[0032] Referring now to FIGURE 7, there is illustrated a flowchart depicting the duplex operation with a face down output, which is initiated at a block 284 and then proceeds to a decision block 286 to determine if the value of N is even. If so, the program then flows to a function block 288 along the "Y" path to print the pages 2, 4, 6, . . . N. If it was determined that the value of N is odd, the program would flow along an "N" path to a function block 290 to print the pages 2, 4, 6, . . ., N-1 ~~2, 4, 6, . . . 1~~. The program 288 would flow to a decision block 294, which determines if N is odd and, if not, flows along a "N" path to the output of function block 290, the output of a decision block 294 is input to function block 290. The output of function block 290 flows through a function block 296, as well as the output along the "N" path of decision block 294. Decision block 296 indicates the manual operation wherein the user flips the output stack without turning it 180° and then inputs it back into the drawer of the printer from which it was obtained. The program will then flow to a decision block 298 to determine if the value of N is even. If so, the program flows along a "Y" path to a function block 300 and the pages 1, 3, 5, . . ., N-1 ~~1, 3, 5, . . . 1~~ and then to the input of a decision block 302. If the value of N is odd, the program flows along the "N" path from decision block 298 to the output of decision block 308 and to a function block 306 to print the pages 1, 3, 5, . . . N. The output of the decision block 302 along the "Y" path also flows to the function block 306 when N is even, and the flowchart flows along the "N" path to an "END" block 310, this being the path from the function block 306.

Please replace paragraph **[0075]** with the following amended paragraph:

[0075] In order to understand the engine allocation, the concept of "job stacks" will be further elaborated upon. A job stack is basically a dynamically sized array of instructions which is delivered to the PPE 426. Each entry in the array is defined by a number of variables. There is a Number of Copies variable, which instructs the PPE 426 to print each page a defined number of times before moving on to the next page. There is a Begin Page variable to define when the first page of a document is to print, and an End Page variable which defines the last page of the document for this instruction. A Command variable is also input, which is utilized for special commands, such as printing separation pages. In the following example in Table 6, a Command value of "0" indicates nothing, while a Command value of "1" indicates that a job separator page must be printed. Job stacks are dynamic and can contain any number of entries. The example in Table 6 utilizes a ten-page document that is to be printed with four copies utilizing three engines, with a separator page between jobs feature turned on. Table 6 is as follows:

Table 6

	PPE1	PPE2	PPE3
Instruction 1	1, 1, 10, 0	1, 4, 10, 0	1, 7, 10, 0
Instruction 2	1, 1, 3, 0	1, 1, 6, 0	1, 1, 10, 0
Instruction 3	1, 0, 0, 1	1, 0, 0, 1	1, 0, 0, 1

Examining the above Table 6, it can be seen that PPE1 will print one copy of pages 1-10, followed by one copy of pages 1-3 of a document followed by a job separator. PPE2 will print one copy of pages 4-10, followed by one copy of pages <u>1-6</u> [[4-6]] followed by a job separator. PPE3 will print one copy of pages 7-10, followed by one copy of pages 1-10, followed by a job separator. Since the printing is face-down printing, once the job is complete, the user simply takes the pages from PPE3 and stacks them on top of the pages from PPE2 (still face down), and then takes this new stack and places it on top of the pages from PPE1 for a completed and collated job.

Please replace paragraph **[0091]** with the following amended paragraph:

[0091] Referring now to FIGURE [[24]] 23, there is illustrated a block diagram for the print adapter 524. The cable 522 is interfaced through a driver/receiver 550 to a FIFO 552. The FIFO 552 is operable to provide an elastic storage capability which is then input to an internal data bus 554. The internal data bus 554 then interfaces with an unpacker/unloader 556, which is operable to retrieve the data from the FIFO 552 and then decompress this data. The entire operation is controlled by a CPU 560, which CPU 560 is operable to control the number of bits per pixel in the unpacking operation. This is then input to a transform block 558, which transform block 558 is operable to perform a calibration adjustment. As will be described hereinbelow, the engines in a given virtual printer are "color balanced." In order to do this, each engine is calibrated and compared to an internal master color space. The data that is transferred to the FIFO 552 is formatted in this master color space. Any aberrations of the printer due to parameters associated with a given engine that may yield to wear, etc., can be compensated for in this calibration procedure. Once calibration is complete, a look-up table 562 is loaded with calibration information, which calibration information is then utilized by the transform block 558 to correct the color space. This data is then input to the marking engine 514.

Please replace paragraph **[0106]** with the following amended paragraph:

[0106] The above noted color balancing calibration procedure is acceptable for color balancing contone images, but there are some aspects of halftone rendering, for example, bi-level and quad-level techniques, that may provide a problem, since the different levels are achieved by spacing dots over a given area, each of the dots generated at the maximum pixel value. Therefore, there is an additional adjustment that is provided for bi-level and quad-level printing formats. This is an adjustment to the maximum pixel value of 255 that is generated, this defines the maximum density value for the printer. It is noted that the maximum density value may already have been adjusted for and stored in the color map block 618-622 associated with the marking [[etch]] engines 612-616. This is the calibration operation already performed to this point. However, there still exists a problem in that the calibration operation at the maximum density is less than adequate. The reason for this is ~~due to the fact~~ that at maximum density the colorimeters utilized to measure high color density level may suffer errors due to gloss variations. Further, visual inspection at high color density levels is ~~unreliable~~ unreliable, whereas measurement at lower density levels is subject to extrapolation errors. However, ~~it is noted that~~ visual inspection at low color density levels is reliable since the eye can determine color shifts at these low density levels. However, the question is whether the visual inspection at low density levels can be extrapolated to a color correction at a high density level.

Please replace paragraph **[0113]** with the following amended paragraph:

[0113] Referring now to FIGURE 32, there is illustrated a ~~diagrammatic~~ block diagram of the operation for correcting for dot gain in various output devices. Although, as noted above, a dot linearization curve can be generated for a given RIP and a given output device, a problem exists when dealing with multiple printers. This ~~is due to the fact that~~ because each printer or output device will have its own dot linearization curve and it can be distinct. Therefore, by creating a single dot linearization curve for single engine and translating this over multiple engines, there is a strong possibility that there will be a "gray shift" between engines. In FIGURE 32, there is illustrated a single RIP 649, which is described hereinabove as a software RIP. This RIP will create a

stored image which is stored in a memory (not shown). For the purposes of this description, the entire RIP operation and storage operation will be represented by the block 649. There are also provided four print engines (PEs) 651, each being a different engine. The RIP 649 in a halftone operation will be passed through a correction block [[651]] 652 to provide a correction therefor, such that when a 50% RIP is requested, the overall RIP operation constituting the blocks 649 and 651 will result in a different output RIP. Although illustrated as two blocks 649 and 651, in actuality the RIP merely changes the percentage that it RIPS. Once RIPed, this is output to a job distributor 653, which is described as the job manager herein, and then distributed to the various print engines 651. The outputs of the print engine 651 will be measured, typically through a visual measurement, with the use of a colorimeter 655. The colorimeter 655 will determine for each color the percent array for a given patch. This will be input to a linearizer 657 to provide dot linearized curves for each engine for each color, cyan, magenta and yellow. This is then input to an averaging block 659 to average all linearization curves for each color and then input this to the correction block [[651]] 652. The correction block [[651]] 652 in general, provides the necessary offset. As an example, consider that there are four print engines with the 50% level resulting in an actual output of 40%, 42%, 44% and 46%. The average of these four numbers would be 43%. Therefore, the correction block 651 would request the RIP 659 to provide a halftone image of 57% rather than 50%. This would result in a perceived average 50% gray image from the print engines 651. This is to be distinguished from the operation above wherein each print engine had the contone calibration disposed therein. That, of course, was directed toward density, whereas the present system is directed toward the problems with "dot coverage" in halftone job. As noted above, each system will have different parameters which are the result of various inherent properties of the particular print engine and to how it generates a particular dot and whether the dot is generated at the appropriate size to provide the appropriate cover. The RIP 649, of course, assumes that every engine operates as an ideal engine. Since it does not, it is necessary to provide this correction at the RIP level, rather than at the print engine level.

Please replace paragraph **[0114]** with the following amended paragraph:

[0114] In certain situations, there are output devices that have dot linearization curves that are outside of the range. If this engine were utilized in the averaging operation, it would essentially pull all the other engines off to some extent. In this case, an aberrant engine would be determined in the averaging operation and then a different operation provided thereby. It has been noticed through ~~impirical~~ <u>empirical</u> tests that the print engine will have a varying dot gain or linearization curve as a function of the density control on the engine. In the preferred embodiment, a Canon 550 engine is utilized which has a software density control. This is facilitated by inputting to the engine a software value which can be stored in the engine, this being a capability of the engine. By varying this software density control, the linearization curve can be varied. Rather than utilize the linearization curve that is output from the aberrant engine, the aberrant engine is excluded from the averaging operation, an average determined and an offset therefrom determined. This offset is utilized to drive the aberrant engine with the density control then utilized to vary the linearization curve thereof to bring it within the bounds of the non-aberrant engine. This is facilitated by density control 661 which receives from the averaging information the information as to which engine is the aberrant engine and then when the correction is applied in the block [[651]] <u>652</u>, the density control value is applied to that engine. This can be done on a job basis and even on a page basis when there is a mixture of contone and halftone.